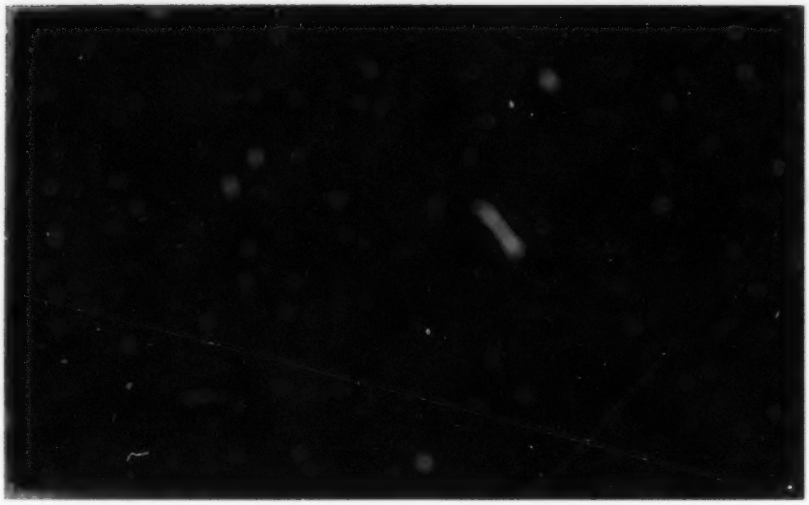
CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE : (514) 343-7575
EMAIL : crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX) : (514) 343-7121

92-07820/

CRT-843

An Exact Algorithm for Multicommodity Location with Balancing Requirements

by

Bernard Gendron[1]
Teodor Gabriel Crainic[1,2]

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Canada H3C 3J7.

[2] Département des sciences administratives, Université du Québec à Montréal, C.P. 6192, succursale A, Montréal, Canada H3C 4R2.

Centre de recherche sur les transports — Publication # 843

August 1992

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 1992

ABSTRACT

The multicommodity location problem with balancing requirements is related to one of the major logistics issues faced by distribution and transportation firms: the management of a fleet of vehicles over a medium to long-term planning horizon. We present a branch-and-bound algorithm in which tight bounds are computed by a dual-ascent procedure. We particularly emphasize the development of efficient branching rules and strong fathoming criteria. The algorithm was tested on a wide variety of randomly generated problems and on a large-scale application to the planning of the land operations of a heterogeneous container fleet. Results show that the algorithm is highly efficient and outperforms other existing methods.

Keywords: branch-and-bound, dual-ascent, multicommodity location with balancing requirements.

RÉSUMÉ

Le problème de localisation multiproduit avec exigences d'équilibrage est relié à un des aspects importants de la planification logistique des entreprises de transport et de distribution, soit la gestion d'une flotte de véhicules à moyen et long termes. Nous présentons un algorithme de "branch-and-bound" qui calcule des bornes au moyen d'une méthode d'ascension duale. Nous nous concentrons particulièrement sur la conception de règles de branchement et de critères de suspension de fouille efficaces. L'algorithme a été testé sur une grande variété de problèmes générés aléatoirement et sur une application de grande taille au problème de la planification du transport terrestre de conteneurs. Les résultats montrent que l'algorithme est extrêmement efficace et qu'il l'emporte sur d'autres méthodes.

Mots-clés: "branch-and-bound", méthode d'ascension duale, problème de localisation multiproduit avec exigences d'équilibrage.

INTRODUCTION

The multicommodity location problem with balancing requirements has been formally introduced by Crainic, Dejax and Delorme (1989), in relation to the management of the land distribution and transportation system of international container shipping companies. Similar problems arise in several other industrial production planning environments.

Crainic, Delorme and Dejax (1992) show that standard location methods are not efficient for this problem, but their branch-and-bound algorithm cannot prove the optimality in a reasonable amount of time, except for simple data instances. Consequently, heuristic methods have been tried to solve the problem. Crainic and Delorme (1992) propose a dual-ascent procedure which generally produces very tight bounds, but still can exhibit large gaps on particular instances. An adaptation of the Tabu Search metaheuristic (Crainic, Gendreau, Soriano and Toulouse, 1992) generally obtains better solutions, but requires a rather large computing time.

In this paper, we present an extremely efficient exact algorithm. It is a branch-and-bound approach which combines an improved dual-ascent bounding procedure, with sharp branching rules and strong fathoming criteria.

In Section I, we formulate the model and derive lower bounds from Lagrangean relaxations of the problem. These bounds define a dual-ascent procedure, similar to the one proposed by Crainic and Delorme (1992). Section II presents the branch-and-bound algorithm. Several experiments were conducted both on a wide variety of randomly generated problems and on data from a large scale application. The computational results of these experiments are reported and analyzed in Section III.

I. FORMULATION AND BOUNDING PROCEDURE

The formulation of the problem is motivated by the following industrial application, related to the management of an international container shipping company. Once a ship arrives at port, the company has to deliver loaded containers, which may come in several types and sizes, to designated in-land destinations. Following their unloading by the importing customer, empty containers are moved to a depot. From there, later on, they may be delivered to customers who request containers for subsequent shipping of their own products. Furthermore, containers may be repositioned to other depots. These interdepot movements are a consequence of the regional unbalances in empty container availabilities and needs throughout the network: some areas lack containers of certain types, while others have surpluses of them. This requires balancing movements of empty containers among depots, and thus differentiates this problem from a classical location/allocation application.

The general problem is therefore to locate depots in order to collect the supply of empty containers available at customers' sites and to satisfy the customer requests for empty containers, while minimizing the total operating costs: the costs of opening and operating the depots, plus the transportation costs between customers and depots and the costs generated by interdepot movements. Note that, in practice, any formulation of this problem may have to be repeatedly solved since, often, container shipping companies do not build their own depots, but rather use facilities from other modes (ports and railyards, mostly).

Crainic, Dejax and Delorme (1992) have already proposed a *static, deterministic, uncapacitated* and *linear* model, which we adapt here to specify a network formulation

of the problem:

Problem P1

Minimize

$$\sum_{j\in D} f_j y_j + \sum_{p\in P}\left(\sum_{(i,j)\in E_1(p)} c_{ijp}x_{ijp} + \sum_{(j,i)\in E_2(p)} c_{jip}x_{jip} + \sum_{(j,k)\in E_3(p)} s_{jkp}w_{jkp}\right) \quad (1)$$

Subject to

$$\sum_{j\in D_1^i(p)} x_{ijp} = O_{ip} \qquad \forall i \in C_1(p), p \in P \quad (2a)$$

$$\sum_{j\in D_2^i(p)} x_{jip} = D_{ip} \qquad \forall i \in C_2(p), p \in P \quad (2b)$$

$$\sum_{i\in C_1^j(p)} x_{ijp} + \sum_{k\in D_1^j(p)} w_{kjp} - \sum_{i\in C_2^j(p)} x_{jip} - \sum_{k\in D_2^j(p)} w_{jkp} = 0 \quad \forall j \in D, p \in P \quad (3)$$

$$x_{ijp} \le O_{ip} y_j \qquad \forall j \in D, i \in C_1^j(p), p \in P \quad (4a)$$

$$x_{jip} \le D_{ip} y_j \qquad \forall j \in D, i \in C_2^j(p), p \in P \quad (4b)$$

$$x_{ijp} \ge 0 \qquad \forall (i,j) \in E_1(p), p \in P \quad (5a)$$

$$x_{jip} \ge 0 \qquad \forall (j,i) \in E_2(p),\ p \in P \quad (5b)$$

$$w_{jkp} \ge 0 \qquad \forall (j,k) \in E_3(p), p \in P \quad (6)$$

$$y_j \in \{0,1\} \qquad \forall j \in D \quad (7)$$

where

P	:	set of products (types of containers);
D	:	set of depots;
$C_1(p)$	:	set of supply nodes (customers) for product p;

$C_2(p)$: set of demand nodes (customers) for product p;

$E_1(p)$: set of customer-depot arcs for product p;

$E_2(p)$: set of depot-customer arcs for product p;

$E_3(p)$: set of depot-depot arcs for product p;

$D_1^i(p) = \{j \in D/(i,j) \in E_1(p)\}$;

$D_2^i(p) = \{j \in D/(j,i) \in E_2(p)\}$;

$D_1^j(p) = \{k \in D/(k,j) \in E_3(p)\}$;

$D_2^j(p) = \{k \in D/(j,k) \in E_3(p)\}$;

$C_1^j(p) = \{i \in C_1(p)/(i,j) \in E_1(p)\}$;

$C_2^j(p) = \{i \in C_2(p)/(j,i) \in E_2(p)\}$;

f_j : fixed cost of opening and operating depot j ($f_j > 0$);

c_{ijp}, c_{jip} : unit transportation costs for product p, between customer i and depot j,

s_{jkp}, s_{kjp} : unit transportation costs for product p, between depots j and k;

O_{ip} : supply of product p at node (customer) i;

D_{ip} : demand of product p at node (customer) i;

y_j : decision variable for depot j,

$$y_j = \begin{cases} 1, \text{ if depot } j \text{ is open} \\ 0, \text{ otherwise} \end{cases}$$

x_{ijp}, x_{jip} : flows of product p between customer i and depot j;

w_{jkp}, w_{kjp} : flows of product p between depots j and k.

Constraints (2) ensure that supply and demand requirements are met, while constraints (3) correspond to flow conservation conditions for the interdepot balancing traffic. Constraints (4) forbid movements through closed depots. Note that analogous constraints for interdepot flows are redundant, if interdepot costs satisfy a triangle inequality, which is assumed throughout the text. For a proof of this, see Crainic, Dejax and Delorme (1992).

In order to compute bounds, we define two subproblems derived by using Lagrangean relaxation methods. But first, we introduce the dual of the formulation, to subsequently establish relationships between Lagrangean multipliers and dual variables.

The *strong relaxation* of problem P1 is obtained by replacing the integrality constraints (7) by $y_j \geq 0$, $\forall j \in D$. The dual of this linear relaxation is:

Problem P2

Maximize

$$\sum_{p \in P} \left(\sum_{i \in C_1(p)} O_{ip}\mu_{ip} + \sum_{i \in C_2(p)} D_{ip}\nu_{ip} \right) \tag{8}$$

Subject to

$$\mu_{ip} + \lambda_{jp} - \gamma_{ijp} \leq c_{ijp} \quad \forall (i,j) \in E_1(p), p \in P \tag{9a}$$

$$\nu_{ip} - \lambda_{jp} - \gamma_{jip} \leq c_{jip} \quad \forall (i,j) \in E_2(p), p \in P \tag{9b}$$

$$\lambda_{kp} - \lambda_{jp} \leq s_{jkp} \qquad \forall (j,k) \in E_3(p), p \in P \tag{10}$$

$$\sum_{p \in P} \left(\sum_{i \in C_1^j(p)} O_{ip}\gamma_{ijp} + \sum_{i \in C_2^j(p)} D_{ip}\gamma_{jip} \right) \leq f_j \qquad \forall j \in D \tag{11}$$

$$\gamma_{ijp} \geq 0 \qquad \forall (i,j) \in E_1(p), p \in P \tag{12a}$$

$$\gamma_{jip} \geq 0 \qquad \forall (j,i) \in E_2(p), p \in P \tag{12b}$$

where $\mu_{ip}$and ν_{ip} are the dual variables associated with constraints (2a) and (2b), while λ_{jp} are associated with constraints (3) and $\gamma_{ijp}[\gamma_{jip}]$ are associated with constraints (4a), [(4b)].

A first subproblem is obtained by relaxing constraints (4a) and (4b) and by introducing them into the objective function with nonnegative Lagrangean multipliers γ_{ijp} and γ_{jip}. We then have the following problem:

Problem P3

Minimize

$$\sum_{j\in D}\left\{f_j-\sum_{p\in P}\left(\sum_{i\in C_1^j(p)}O_{ip}\gamma_{ijp}+\sum_{i\in C_2^j(p)}D_{ip}\gamma_{jip}\right)\right\}y_j+$$

$$\sum_{p\in P}\left\{\sum_{(i,j)\in E_1(p)}(c_{ijp}+\gamma_{ijp})x_{ijp}+\sum_{(j,i)\in E_2(p)}(c_{jip}+\gamma_{jip})x_{jip}+\sum_{(j,k)\in E_3(p)}s_{jkp}w_{jkp}\right\} \quad (13)$$

Subject to (2a), (2b), (3), (5a), (5b), (6), (7).

One of the most delicate operations of using Lagrangean relaxation to compute bounds for enumeration algorithms usually consists in obtaining values for the Lagrangean multipliers which ensure the tightness of the bounds. In our case, however, this may easily be achieved. Indeed, for problem P3, when the γ multipliers are fixed, the first term of (13) may be rewritten as

$$\sum_{j\in D}\min\left\{0,\ f_j-\sum_{p\in P}\left(\sum_{i\in C_1^j(p)}O_{ip}\,\gamma_{ijp}+\sum_{i\in C_2^j(p)}D_{ip}\,\gamma_{ijp}\right)\right\} \quad (14)$$

and constraints (7) can be dropped. One may then easily observe that, in order to obtain the best possible lower bound on the optimal objective value of the original problem, the γ multipliers should be restricted to values satisfying (11), the constraints linking the fixed costs to the γ variables in the dual formulation of the strong relaxation. With such values for γ, the first term of the objective function (13) can be dropped, and we obtain the following equivalent problem:

Problem P4

Minimize

$$\sum_{p\in P}\left\{\sum_{(i,j)\in E_1(p)}(c_{ijp}+\gamma_{ijp})x_{ijp}+\sum_{(j,i)\in E_2(p)}(c_{jip}+\gamma_{jip})x_{jip}+\sum_{(j,k)\in E_3(p)}s_{jkp}w_{jkp}\right\} \quad (15)$$

Subject to (2a), (2b), (3), (5a), (5b), (6).

Problem P4 is a *multicommodity uncapacitated minimum cost flow problem*, which decomposes into $|P|$ independent uncapacitated minimum cost flow problems. It can then be solved by any of the known methods for this class of problems.

From the optimal solution $(\overline{x},\overline{w})$ of problem P4, an upper bound is easily obtained by setting

$$\overline{y}_j=\left\{\begin{array}{ll}0, & \text{if } \overline{x}_{ijp}=\overline{x}_{jip}=0 \quad \forall (i,j)\in E_1,\ (j,i)\in E_2(p), p\in P\\ 1, & \text{otherwise}\end{array}\right\}\forall\, j\in D \quad (16)$$

A second subproblem of P1 may be derived by relaxing the balancing constraints (3) and by introducing them into the objective function with multipliers $(-\lambda_{jp})$:

Problem P5

Minimize

$$\sum_{j\in D}f_j\,y_j+\sum_{p\in P}\left\{\sum_{(i,j)\in E_1(p)}(c_{ijp}-\lambda_{jp})\,x_{ijp}+\sum_{(j,i)\in E_2(p)}(c_{ijp}+\lambda_{jp})\,x_{jip}+\sum_{(j,k)\in E_3(p)}\left(s_{jkp}-\{\lambda_{kp}-\lambda_{jp}\}\right)w_{jkp}\right\} \quad (17)$$

Subject to (2a), (2b), (4a), (4b), (5a), (5b), (6), (7).

Note that the w variables are restricted to be nonnegative and are not part of any other constraint besides the balancing ones. Then, if the λ multipliers are restricted

to values satisfying the dual constraints (10), we obtain a lower bound on problem P5, and hence on the original problem by solving:

Problem P6

Minimize

$$\sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{(i,j) \in E_1(p)} (c_{ijp} - \lambda_{jp}) x_{ijp} + \sum_{(j,i) \in E_2(p)} (c_{jip} + \lambda_{jp}) x_{jip} \right\} \quad (18)$$

Subject to (2a), (2b), (4a), (4b), (5a), (5b), (7).

Problem P6 is a *simple plant location problem* (Krarup and Pruzan, 1983), also called simple uncapacitated facility location problem (Aikens, 1985). Here, the multicommodity aspect of the problem, as well as the fact that customers may both send to and receive from depots, does not increase the difficulty of the problem. It does, however, affect the implementation of a solution procedure and it increases data storage requirements.

One of the most efficient methods for solving the simple plant location problem (see Krarup and Pruzan, 1983; and Cornuejols, Nemhauser and Wolsey, 1990; for complete surveys) is the DUALOC algorithm proposed by Erlenkotter (1978). The algorithm works on a condensed dual of the strong relaxation of P6 in which the γ variables do not have to be explicitly computed. From any dual solution, the algorithm easily derives a primal solution satisfying the integrality constraints; the couple of primal-dual solutions thus obtained may, however, violate some of the complementary slackness conditions. Through a *dual-ascent procedure*, which starts with some initial dual solution, the algorithm incrementally adjusts the dual variables in such a way that complementary slackness violations are reduced. The algorithm also incorporates an

adjustment procedure that tries to increase the dual objective value through decreasing some dual variables while increasing others. At the end of the process, a branch-and-bound procedure is initiated in case the optimal solution has not already been found. For complete details of the method, we refer the reader to the papers of Erlenkotter (1978) and Van Roy and Erlenkotter (1982). For our purposes, it is sufficient to recall that the initial dual solution must satisfy the following conditions:

$$\mu_{ip} \geq \min_{j \in D_1^i(p)} \{c_{ijp} - \lambda_{jp}\}, \qquad \forall i \in C_1(p), p \in P \tag{19a}$$

$$\nu_{ip} \geq \min_{j \in D_2^i(p)} \{c_{jip} + \lambda_{jp}\}, \qquad \forall i \in C_2(p), p \in P \tag{19b}$$

Furthermore, since the bounds computed by DUALOC are very tight, we use the method as a heuristic and do not incorporate the branch-and-bound procedure. Thus, in the remainder, when we refer to DUALOC, we mean the heuristic, not the exact algorithm.

An upper bound on the original problem P1 may easily be computed starting from a primal solution, obtained by DUALOC, which satisfies the integrality conditions. Let $\overline{y}$ be the realization of the vector of decision variables associated with that primal solution. We can then solve the following multicommodity uncapacitated minimum cost flow problem, which also decomposes into $|P|$ independent uncapacitated minimum cost flow problems:

Problem P7

Minimize

$$\sum_{p \in P} \left(\sum_{(i,j) \in E_1(p)} c_{ijp} x_{ijp} + \sum_{(j,i) \in E_2(p)} c_{jip} x_{jip} + \sum_{(j,k) \in E_3(p)} s_{jkp} w_{jkp} \right) \tag{20}$$

Subject to (2a), (2b), (3), (5a), (5b), (6)

$$x_{ijp} \leq O_{ip}\,\overline{y}_j \qquad \forall j \in D,\ i \in C_1^j(p), p \in P \tag{21a}$$

$$x_{jip} \leq D_{ip}\,\overline{y}_j \qquad \forall j \in D,\ i \in C_2^j(p), p \in P \tag{21b}$$

In summary, we may state two bounding procedures derived from the Lagrangean relaxation subproblems P3 and P5:

Procedure FLIP (γ) (Multicommodity uncapacitated minimum cost flow problem)

1. Compute a lower bound Z^L by solving problem P4.
2. Compute an upper bound Z^U by using (16).

Procedure FLOP (λ) (Simple plant location problem)

1. Compute a lower bound Z^L by applying DUALOC to problem P6.
2. Compute an upper bound Z^U by solving problem P7.

Executing the two procedures iteratively, using as input to one the multipliers generated when solving the other, produces an increasing sequence of lower bounds. To see this, observe first that any feasible dual solution of both problems P4 and P6, is also feasible for the dual problem P2, since the multipliers are fixed to values satisfying the dual constraints. Note also that problem P4 is solved to optimality, while only a lower bound is computed for problem P6 through a dual-ascent procedure (DUALOC). Hence, if the optimal dual solution of problem P4 is used as input for DUALOC, the dual objective increases. Consequently, it only remains to be shown that the dual optimal solution (μ, ν, λ) of problem P4 satisfies conditions (19a) and (19b). We state this formally as:

Proposition 1.

Let (μ, ν, λ) *be a dual optimal solution of problem P4. We then have*

$$\mu_{ip} \geq \min_{j \in D_1^i(p)} \{c_{ijp} - \lambda_{jp}\}, \qquad \forall i \in C_1(p), p \in P \tag{19a}$$

$$\nu_{ip} \geq \min_{j \in D_2^i(p)} \{c_{jip} + \lambda_{jp}\}, \qquad \forall i \in C_1(p), p \in P \tag{19b}$$

To prove this, suppose, without loss of generality, that there exists $i \in C_1(p)$, $p \in P$ (i.e. $O_{ip} > O$), such that $\mu_{ip} < \min_{j \in D_1^i(p)} \{c_{ijp} - \lambda_{jp}\} = c_{ij^*p} - \lambda_{j^*p}$ Since the γ multipliers are nonnegative, $\mu_{ip} < c_{ij^*p} - \lambda_{j^*p} + \gamma_{ij^*p}$. Then, there exists $\xi_{ip} > 0$ such that $\mu_{ip} + \xi_{ip} = c_{ij^*p} - \lambda_{j^*p} + \gamma_{ij^*p}$. Consequently, by setting $\overline{\mu}_{ip} = \mu_{ip} + \xi_{ip}$, we obtain a feasible solution for the dual of problem P4 and the dual objective strictly increases by a quantity $O_{ip}\, \xi_{ip} > 0$. But, this contradicts the fact that (μ, ν, λ) is a dual optimal solution of problem P4.

We may now formally state a dual-ascent procedure which, given the previous arguments, is similar to the one proposed by Crainic and Delorme (1992), with some improvements that we discuss later. Define

iter	=	number of iterations performed;
$last_Z^L$	=	the lower bound generated at the previous iteration;
$best_Z^U$	=	the best upper bound on the original problem generated so far.

Procedure FLIP-FLOP

1. Initialize γ and $last_Z^L$; $best_Z^u \leftarrow +\infty$; $iter \leftarrow 1$.
2. If ($iter$ mod $2 = 1$), then

call procedure FLIP (γ);
compute (μ, ν, λ);

else

call procedure FLOP(λ);
compute γ.

3. $best_Z^u \leftarrow \min\{best_Z^u, Z^U\}$
4. If $\frac{best_Z^u - Z^L}{Z^L} < \epsilon_1$ or $\frac{Z^L - last_Z^L}{last_Z^L} < \epsilon_2$ or $iter = maxiter$, then STOP: $best_Z^U$ is the best known value on the objective of the original problem.
5. $last_Z^L \leftarrow Z^L$; $iter \leftarrow iter + 1$; go to step 2.

A few remarks are in order to fully understand the procedure:

1. The procedure starts by solving first a FLIP subproblem, since this choice has been experimentally proven to be superior to the alternative (Crainic and Delorme, 1992). Indeed, when solving first a FLOP subproblem, one does not take into account the influence of the balancing flows. In particular, some depots may be closed even if they might be required in order to satisfy the balancing constraints.
2. Contrary to Crainic and Delorme, we do not compute upper bounds on the strong relaxation, since we observed that it has no significant effect on the convergence of the procedure.
3. Crainic and Delorme define an iteration as the resolution of two successive subproblems. Consequently, they verify stopping criteria only when two new subproblems have been solved. When embedded into a branch-and-bound algorithm, our procedure is faster since it tests stopping criteria each time a subproblem is treated.

4. There are three stopping criteria. The first stops the procedure when the relative gap between the best upper bound and the lower bound is sufficiently small, as measured by $\epsilon_1 > 0$. The second comes into play when the lower bound has not significantly increased, as measured by $\epsilon_2 > 0$, compared to its value at the previous iteration. The third criterion terminates the procedure when the number of iterations performed attains a given maximum, *maxiter*.

5. When computing (or initializing) the γ multipliers, we use the *gamma adjustment rule*. It consists of removing some proportion of the slack that may appear in the dual constraints (11) after the resolution of a FLOP subproblem (or initially). Thus, the lower bound computed when solving a FLIP subproblem may be increased.

   Formally, define the slack variables in constraints (11) as

$$e_j = f_j - \sum_{p \in P} \left(\sum_{i \in C_1^j(p)} O_{ip} \gamma_{ijp} + \sum_{i \in C_2^j(p)} D_{ip} \gamma_{jip} \right) \qquad \forall j \in D \qquad (22)$$

   The vector of adjusted variables, γ^a, is then computed by

$$\gamma^a_{ijp} = \gamma_{ijp} + \theta_j e_j / \Delta_j \qquad \forall (i,j) \in E_1(p), p \in P \qquad (23a)$$

$$\gamma^a_{jip} = \gamma_{jip} + \theta_j e_j / \Delta_j \qquad \forall (j,i) \in E_2(p), p \in P \qquad (23b)$$

   where

$$\Delta_j = \sum_{p \in P} \left(\sum_{i \in C_1^j(p)} O_{ip} + \sum_{i \in C_2^j(p)} D_{ip} \right) \qquad \forall j \in D \qquad (24)$$

$$0 \leq \theta_j \leq 1 \qquad \forall j \in D \qquad (25)$$

In particular, $\theta_j = 0, \forall j \in D$ (*minimum adjustment rule*) corresponds to solving the FLIP subproblem with the multipliers generated by the last FLOP iteration (or the initial multipliers). On the other hand, $\theta_j = 1, \forall j \in D$ (*maximum adjustment rule*) is equivalent to removing any slack in constraints (11); this was used as a definition of the gamma adjustment rule in Crainic and Delorme (1992). See Section III for a discussion on the performance of the rule.

The initialization of the procedure (step 1) is discussed in the next section, which presents the branch-and-bound algorithm.

II. BRANCH-AND-BOUND ALGORITHM

The general idea of the branch-and-bound algorithm is to try to solve the original problem P1, and in case of an unsuccessful attempt, to decompose it into easier subproblems. These subproblems are further divided, unless their solution is found or it is determined that they cannot lead to an optimal solution for the original problem. More precisely, the decomposition of any subproblem S consists in choosing a free depot j^*(i.e. $y_{j^*} \in \{0,1\}$), and in creating the subproblems S_0 and S_1, each having an additional constraint, respectively $y_{j^*} = 0$ and $y_{j^*} = 1$. This decomposition process defines a binary tree, each node being associated to a subproblem S, which is entirely characterized by the following sets:

$$
\begin{aligned}
D_0(S) &= \{j \in D / y_j = 0\} : \text{set of closed depots;} \\
D_1(S) &= \{j \in D / y_j = 1\} : \text{set of open depots;} \\
D_{01}(S) &= \{j \in D / y_j \in \{0,1\}\} : \text{set of free depots.}
\end{aligned}
$$

The subproblem S_R at the root node of the tree corresponds to the original problem; hence, in this case, all depots are free ($D_0(S_R) = D_1(S_R) = \emptyset$ and $D_{01}(S_R) = D$). Furthermore, using the terminology of trees, we name S_0 and S_1, the *0-son node* and the *1-son node*, respectively, of the *father node* S.

Formally, the algorithm keeps a list L of generated subproblems and a value *best_Z* of the best solution identified thus far. It proceeds as follows:

Procedure BB

1. $L \leftarrow \{S_R\}$; $best_Z \leftarrow +\infty$.
2. ***Selection rule***: select a subproblem S in L and delete it from L.
3. Compute a lower bound Z^L and an upper bound Z^U for S.

4. $best_Z \leftarrow \min\{best_Z, Z^U\}$.

5. ***Branching rule***: if $Z^L < best_Z$, then choose $j^* \in D_{01}(S)$ and add to L the two subproblems S_i $(i = 0, 1)$ defined by the sets

$$D_i(S_i) = D_i(S) \cup \{j*\}$$

$$D_{(i+1)mod2}(S_i) = D_{(i+1)mod2}(S)$$

$$D_{01}(S_i) = D_{01}(S) - \{j*\}$$

6. If $L = \emptyset$, then STOP: $best_Z$ is the value of the optimal solution of the original problem.

7. Go to step 2.

The performance of the algorithm is mainly influenced by three factors: the tightness of the bounds, the ability to avoid unnecessary computations and the way subproblems are generated and selected. In the remainder of the section, we show, for each of these factors, how the structure of the FLIP-FLOP bounding procedure can be exploited to improve the efficiency of procedure BB.

For any subproblem S, the bounds are computed by defining a new formulation of problem P1, where the fixed costs f_j are replaced by the modified fixed costs f_j^{mod} :

$$f_j^{mod} = +\infty \qquad \forall_j \in D_0(S) \quad \textbf{(26a)}$$

$$f_j^{mod} = 0 \qquad \forall_j \in D_1(S) \quad \textbf{(26b)}$$

$$f_j^{mod} = f_j \qquad \forall_j \in D_{01}(S) \quad \textbf{(26c)}$$

A lower bound Z^L for S is then derived by applying procedure FLIP-FLOP to this problem, and by adding to the value of the new objective function the constant

term $\sum_{j \in D_1(S)} f_j$. The modifications needed to compute an upper bound Z^U for S are also easy to handle.

From the information obtained when computing bounds for S, we can derive lower bounds on the optimal value of subproblems S_0 and S_1. In particular, for subproblem S_1, we have the following:

Proposition 2.

Let S be a subproblem selected at step 2 of procedure BB and Z^L its lower bound computed by procedure FLIP-FLOP. Let S_1 be the 1-son node of S and $Z(S_1)$ the optimal value of S_1. Then, if the last iteration of procedure FLIP-FLOP is a FLOP, we have

$$Z^L + e_{j^*} \leq Z(S_1) \tag{27a}$$

while, if the last iteration of procedure FLIP-FLOP is a FLIP, we have

$$Z^L + (1 - \theta_{j^*}) e_{j^*} \leq Z(S_1) \tag{27b}$$

To prove this, observe first that $f_{j^*}^{mod} = 0$ in the formulation of S_1. This implies that the γ variables associated with j^* must all be equal to 0 in any dual solution of S_1. Now, consider $(\mu, \nu, \lambda, \gamma)$, the dual solution of S generated by procedure FLIP-FLOP, and suppose further that the last iteration was a FLOP. Then, by setting

$$\overline{\mu}_{ip} = \mu_{ip} - \gamma_{ij^*p}, \quad \forall i \in C_1^{j^*}(p),\ p \in P$$
$$\overline{\nu}_{ip} = \nu_{ip} - \gamma_{j^*ip}, \quad \forall i \in C_2^{j^*}(p),\ p \in P$$

and by keeping the other variables at their current values, we obtain a feasible solution for the dual of S_1 and its value is

$$Z^L + f_{j^*} - \sum_{p \in P} \left(\sum_{i \in C_1^{j^*}(p)} O_{ip} \gamma_{j^*p} + \sum_{i \in C_2^{j^*}(p)} D_{ip} \gamma_{j^*ip} \right) = Z^L + e_{j^*}$$

Thus, (27a) follows immediately. To show (27b), just recall that the vector of adjusted γ variables, γ^a, is used when solving a FLIP subproblem.

Note that this proposition has similar versions in other Lagrangean-based or dual-ascent algorithms (see for example, Geoffrion 1974; and Balakrishnan, Magnanti and Wong, 1989). It is used here to strengthen the initialization of the FLIP-FLOP bounding procedure and, also, to define sharp fathoming criteria.

The initialization of procedure FLIP-FLOP (step 1) proceeds as follows. At the root node, γ and $last_Z^L$ are set equal to 0. For other nodes, we use the information generated when solving the father node. The γ variables keep their values, except those associated with a newly fixed depot, which take the value 0 in case of an open depot, and $+\infty$ in case of a closed depot. For subproblem S_1, $last_Z^L$ is initialized to $Z^L + e_{j^\bullet}$, if the last FLIP-FLOP iteration at the father node was a FLOP, or to $Z^L + (1-\theta_{j^\bullet})e_{j^\bullet}$, if the alternative is verified. Of course, for subproblem S_0, $last_Z^L$ is initialized to Z^L.

To further tighten the bounds, we use an experimental observation: at the root node, the gap between the bounds may be significantly reduced by solving complete FLIP-FLOP sequences, as in Crainic and Delorme (1992) (see remark 3 at the end of Section I). Consequently, when the last iteration of procedure FLIP-FLOP (at the root node) is a FLIP, we compute an additional FLOP iteration. Note, however, that this modification would be extremely inefficient if implemented for other nodes than the root.

Now, we turn our attention to the second performance factor: the ability of the algorithm to avoid unnecessary computations. We adapted some known techniques

to reduce the size of the tree and to save on computing time. These techniques include *fathoming criteria*, to assist the pruning of the tree, a *feasibility test*, to eliminate the generation of infeasible subproblems, and a *reduction test*, to eliminate the computation of bounds for a certain class of subproblems.

A first obvious fathoming criterion is obtained by adding the following test after the computation of a lower bound Z^L (step 1) in procedures FLIP and FLOP:

F1) If $Z^L \geq best_Z$, then go to step 6 of procedure BB.

Stronger fathoming criteria are derived by application of Proposition 2. A first one is inserted at step 5 of procedure BB:

F2) If ($iter\ mod 2 = 0$ and $Z^L + e_{j^\bullet} \geq best_Z$) or ($iter\ mod 2 = 1$ and $Z^L + (1 - \theta_{j^\bullet})e_{j^\bullet} \geq best_Z$), then do not generate S_1.

A second criterion is tested after selecting a subproblem S at step 2. If S is the 1-son node of a given subproblem S', for which we have kept in memory its lower bound Z^L, the value of the slack variable $e_{j^\bullet}$ and the number of iterations (*iter*) performed when executing procedure FLIP-FLOP on S', then we have:

F3) If ($iter\ mod 2 = 0$ and $Z^L + e_{j^\bullet} \geq best_Z$) or ($iter\ mod 2 = 1$ and $Z^L + (1 - \theta_{j^\bullet})e_{j^\bullet} \geq best_Z$), then go to step 6 of procedure BB.

The feasibility test is another technique used to reduce the size of the tree. At the root node and each time a 0-son node is selected at step 2, it verifies if some depots must be open in order to satisfy supply or demand requirements. In the affirmative, these depots are added to the set of open depots.

The last technique used to avoid unnecessary computations is the reduction test. It is used after selecting a subproblem S at step 2 of procedure BB. If S is the 0-son

node of a given subproblem S', for which we have kept in memory Z^L, *iter* and $\overline{y}_{j^*}$ (as defined by (16)), we then implement the following test:

R1) If (no depot is opened by the feasibility test and $iter\ mod 2 = 1$ and $\overline{y}_{j^*} = 0$), then go to step 5 of procedure BB.

Thus, the reduction test eliminates the computation of bounds for a certain class of subproblems. To see why it holds true, remark that $\overline{y}_{j^*} = 0$ implies that there is no flow circulating through depot j^* in the solution of the last FLIP iteration on S'. Since S is obtained from S' by closing depot j^* only (no depot being opened by the feasibility test), the lower bound that would be computed by solving a FLIP iteration on S must equal the lower bound generated when solving the father node S'. Thus, the FLIP-FLOP procedure would be stopped immediately by the application of the second termination criterion.

The way subproblems are generated and selected makes for a third algorithmic performance factor. Here, the objective is to define efficient selection and branching rules. The efficiency of the rules may be measured by their contribution to rapidly finding good (or ideally, optimal) solutions and their ability to assist the pruning of the tree.

In the present case, the depth-first rule, which selects one of the deepest nodes in the tree, is an obvious choice. Indeed, it is well-known (see, for example, Ibaraki 1987; and Nemhauser and Wolsey, 1988) that this strategy minimizes computer storage. Furthermore, here, it also simplifies the initialization of the bounding procedure and the application of both the fathoming criterion F3 and the reduction test.

We now present several branching rules which aim to answer the following

questions:

1. In case there are two nodes with the same depth, which one should be selected first, the 0-son node or the 1-son node?
2. How do we choose j^* in step 5 of procedure BB?

The rules are divided into two categories: primal rules and dual rules. In the first category, we use the information obtained when computing an upper bound for a given subproblem S. In the second category, the dual variables generated by the computation of the lower bound are used to choose j^*.

X-rules

These primal rules compare the "positive" activity at a depot j, as measured by customer-depot flows circulating through the depot, to the fixed cost for opening and operating depot j. Formally, we use the measure

$$X_j = \left\{ \sum_{p \in P} \left(\sum_{(i,j) \in E_1(p)} \overline{x}_{ijp} + \sum_{(j,i) \in E_2(p)} \overline{x}_{jip} \right) \right\} \Big/ f_j \quad \forall j \in D_{01}(S) \qquad (28)$$

to specify the following two rules:

1. X_0-rule: choose $j^* = \arg \min_{j \in D_{01}(S)} \{X_j\}$ and select first subproblem S_0.
2. X_1-rule: choose $j^* = \arg \max_{j \in D_{01}(S)} \{X_j\}$ and select first subproblem S_1.

Note that similar rules used by Crainic, Delorme and Dejax (1989) in their branch-and-bound algorithm have succeeded in finding rapidly good solutions.

Λ-rules

From the objective function of problem P6 (the FLOP problem), we easily see that relatively large positive values of λ_{jp} imply that depot j tends to attract more flow than it sends to customers. A similar reasoning applies when λ_{jp} are large negative

values. Thus, a large absolute value of λ_{jp} indicates that depot j have important shortages or surpluses of product p, which result in interdepot balancing flows. This forms the rationale behind the measure

$$\Lambda_j = \left(\sum_{p \in P} |\lambda_{jp}| \right) \Big/ f_j \qquad \forall j \in D_{01}(S) \tag{29}$$

and the corresponding two rules:

1. Λ_0-rule: choose $j^* = \arg \min_{j \in D_{01}(S)} \{\Lambda_j\}$ and select first subproblem S_0.
2. Λ_1-rule: choose $j^* = \arg \max_{j \in D_{01}(S)} \{\Lambda_j\}$ and select first subproblem S_1.

Γ-rules

These dual rules compare the fixed cost for opening and operating depot j, to the values of the γ multipliers associated with depot j. Formally, we use the slack variable e_j, as given by (22), to define the following two rules:

1. Γ_0-rule: choose $j* = \arg \max_{j \in D_{01}(S)} \{e_j\}$ and select first subproblem S_0.
2. Γ_1-rule: choose $j* = \arg \min_{j \in D_{01}(S)} \{e_j\}$ and select first subproblem S_1.

The performance of these rules are analyzed in the next section, which presents computational experiments.

III. COMPUTATIONAL EXPERIMENTS

The objective of the numerical experimentation is to study the behavior of the algorithm, particularly with respect to the branching rules, the fathoming criteria and the reduction test. We also want to determine the influence of the gamma adjustment rule on the performance of the method.

We implemented the algorithm in FORTRAN, using the primal simplex code RNET (Girgoriadis and Hsu, 1979) to solve the multicommodity uncapacitated minimum cost flow problems. In order to solve the simple plant location problems, we coded our own multicommodity version of DUALOC (Erlenkotter, 1978). The dual-ascent phase follows Erlenkotter, while the adjustment step is the primal-dual procedure of Van Roy and Erlenkotter (1982). According to them, this procedure is much more efficient than the dual adjustment method proposed by Erlenkotter (1978). Note also that, in our implementation, the adjustment phase is repeated as long as the dual objective value continues to increase.

In all tests, the parameters of procedure FLIP-FLOP were given the following values: $\epsilon_1 = \epsilon_2 = 0.01$ and *maxiter* = 10. Smaller values were tested but the differences were marginal. Note also that the given maximum number of iterations was never achieved. Typically, two or four iterations are performed at the root node (remember that we force the last iteration to be a FLOP), while one to four iterations (but, most of the time, only one) are necessary for other nodes. In almost all cases, the procedure is stopped because the lower bound has not significantly increased compared to its previous value.

The program was compiled with the *f77* compiler, using the –O option, and all experiments were performed on a SUN Sparc2 workstation.

The tests were conducted on two sets of randomly generated problems, and on data based on an actual large scale application. The procedure used to generate random problems is described in Crainic, Delorme and Dejax (1992).

Table I displays the typical dimensions of the randomly generated problems. This type of problem is described by its label: "M" and "L" indicate medium and large-sized problems respectively, n represents the degree of unbalance present in the network (the higher the figure, the larger the required interdepot flows), while p corresponds to the number of products. Note that for a specific problem, the figures indicated in Table I may vary by a few units. For multiproduct problems, the figures for customer-depot and interdepot arcs are multiplied by the number of products. Note also that, for any of these problems, the single-product networks have a similar structure and represent dense graphs.

Table I. Typical dimensions for randomly generated problems

Type of problem	Customers	Candidate depots	Customer-depot arcs	Interdepot arcs
M*Z*n*P*p	125	25	1750	600
L*Z*n*P*p	219	44	5260	1892

The actual application considered the planning of the land operations of a major European-based international container shipping company. The test problem consists of 12 aggregated types of containers (products), 289 aggregated customer zones, 87 candidate depots, in a geographical zone covering five western European countries. Once the 9 initial transportation modes have been aggregated into the least cost (eventually multimodal) arcs between each pair of nodes, the network contains 4197

interdepot arcs and 32733 customer-depot arcs (with a maximum of 746 interdepot arcs and 3620 customer-depot arcs per product). The problem differs significantly from the random ones since its single-product networks do not have the same structure and represent relatively sparse graphs. We label this problem "RP".

We selected six medium-sized problems and six large-sized problems in all experiments. For each problem, we generated two data instances by multiplying the original fixed costs, in one case by 100 and in the other case by 1000. For the RP problem, we generated four data instances: for one of them, we kept the original fixed costs, while in the other cases, we multiplied the fixed costs by 10, 100 and 1000. The objective of these modifications is to study the sensibility of the method to a change in fixed costs.

The first experiment was dedicated to determining the relative efficiency of the branching rules presented in Section II. For each data instance, we tested the six branching rules, executing the branch-and-bound algorithm up to 1000 iterations. The minimum gamma adjustment rule was used in all tests. Other adjustments display the same general behavior with respect to the relative performance of the branching rules.

Table II shows, for each problem and each branching rule, the number of iterations performed ("Total nodes"). When the tree is not completely explored, a "+" indicates that the optimal solution was found, while a "-" denotes the contrary.

Table II. Tests on branching rules (Total nodes)

Problem	X_0	X_1	Λ_0	Λ_1	Γ_0	Γ_1
MZ1P1 (x100)	263	389	267	205	326	1000$^+$
MZ1P1 (x1000)	161	816	157	936	84	1000$^+$
MZ1P4 (x100)	63	733	61	289	70	877
MZ1P4 (x1000)	390	1000$^+$	465	503	116	1000$^-$
MZ2P3 (x100)	599	1000$^-$	592	1000$^-$	695	1000$^+$
MZ2P3 (x1000)	45	110	44	64	31	153
MZ2P6 (x100)	316	682	326	705	565	1000$^+$
MZ2P6 (x1000)	354	1000$^-$	354	1000$^-$	628	1000$^+$
MZ4P1 (x100)	318	270	328	219	112	383
MZ4P1 (x1000)	120	330	119	271	188	742
MZ4P4 (x100)	282	257	268	273	195	460
MZ4P4 (x1000)	707	1000$^-$	684	504	787	1000$^+$
LZ1P1 (x100)	146	1000$^+$	169	712	114	1000$^+$
LZ1P1 (x1000)	513	1000$^-$	556	1000$^-$	1000$^+$	1000$^-$
LZ1P2 (x100)	1000$^-$	1000$^-$	1000$^-$	1000$^-$	1000$^-$	1000$^-$
LZ1P2 (x1000)	100	332	108	178	60	1000$^+$
LZ3P1 (x100)	370	1000$^-$	393	1000$^-$	431	1000$^-$
LZ3P1 (x1000)	226	282	227	453	64	1000$^-$
LZ3P2 (x100)	1000$^-$	1000$^-$	1000$^-$	1000$^-$	1000$^+$	1000$^-$
LZ3P2 (x1000)	1000$^-$	1000$^-$	1000$^-$	1000$^-$	1000$^+$	1000$^-$
LZ5P1 (x100)	1000$^+$	1000$^-$	1000$^+$	1000$^-$	1000$^+$	1000$^-$
LZ5P1 (x1000)	498	1000$^-$	488	1000$^-$	200	1000$^-$
LZ5P2 (x100)	1000$^-$	1000$^-$	1000$^-$	1000$^-$	1000$^+$	1000$^-$
LZ5P2 (x1000)	1000$^+$	1000$^+$	1000$^+$	1000$^+$	1000$^+$	1000$^+$
RP	201	114	201	149	51	165
RP (x10)	585	361	585	509	253	626
RP (x100)	1000$^+$	837	1000$^+$	1000$^-$	260	499
RP (x1000)	228	163	228	159	89	128

Several conclusions emerge from the experiment:

- The rules that select first a 1-son node are generally inefficient, being outperformed most of the time by the corresponding rules that branch first to a 0-son node.
- The X_0 and Λ_0 rules behave almost identically, since they both measure the "importance" of a depot in terms of the flow component of the problem. However, the Λ_0-rule should be preferred since it is computationally less expensive.
- The Γ_0-rule is clearly superior to the others, when applied to solve large-sized and RP problems. On medium-sized problems, the rules X_0 and Λ_0 are good competitors. However, even for these problems, we observed that generally the Γ_0-rule finds the optimal solution earlier than the other rules. The superiority of the Γ_0-rule may be easily explained. On the one hand, the rule defines an appropriate direction for the research of an optimal solution. Indeed, since the free depot selected by the rule has the largest slack value, it has a good chance of being closed in an optimal solution. On the other hand, the rule combines powerfully with the fathoming criteria to reduce the size of the tree. In fact, the 1-son node generated by the rule has a good probability of being either immediately fathomed by criterion F2, or subsequently by criterion F3. Our general conclusion therefore is that the Γ_0-rule is preferable and should be used in all cases.

In order to examine the performance of the algorithm with respect to the gamma adjustment rule, we conducted several experiments with various values for parameters θ_j. We first observed that the algorithm is very sensible to a positive initial adjustment ($\theta_j > 0, \forall_j \in D$): when the first FLIP subproblem (at the root node) is solved with positive γ values, the method generally performs poorly. On the other hand, when the initial adjustment follows the minimum rule ($\theta_j = 0, \forall_j \in D$), the algorithm performs

well, irrespective of the adjustment rule subsequently used (minimum, maximum or positive). This behavior may be easily understood since, even if it increases the initial lower bound, a positive adjustment introduces a bias in the solution of the minimum cost flow problem; thus, it may subsequently lead to an erroneous treatment of the location component of the problem. Consequently, we recommend to use the minimum adjustment rule when solving the first FLIP subproblem at the root node of the tree.

For subsequent iterations of the bounding procedure and for other nodes than the root, we observed that the best overall adjustments were the extremal ones. On the one hand, the minimum adjustment rule maximizes the number of 1-son nodes fathomed by criterion F2, since all slack variables are maintained at their highest values. On the other hand, the maximum adjustment rule maximizes the number of 0-son nodes for which the bounds are not computed by the application of reduction test R1. This last behavior may be easily explained, since the maximum rule increases significantly the transportation costs in the formulation of the FLIP subproblem. Thus, it yields a better chance of identifying a network flow solution with inactive depots.

Tables III and IV show the results of the tests conducted on the minimum and maximum γ adjustment rules, respectively. The following measures are indicated:

- "Initial Gap (%)": the gap between the bounds computed at the root node;
- "Fathomed Nodes": the number of nodes fathomed by criterion F2;
- "Reduced Nodes": the number of nodes for which the bounds were not computed due to the application of reduction test R1;

Table III. Tests with the minimum gamma adjustment rule

Problem	Initial Gap (%)	Fathomed Nodes	Reduced Nodes	*Best_Z* Node	Total Nodes	Time (CPUs)
MZ1P1 (x100)	0.663	33	2	1	326	56
MZ1P1 (x1000)	6.934	5	3	64	84	13
MZ1P4 (x100)	0.296	9	0	1	70	51
MZ1P4 (x1000)	3.009	19	0	62	116	74
MZ2P3 (x100)	1.126	28	0	181	695	298
MZ2P3 (x1000)	0.777	18	0	1	31	24
MZ2P6 (x100)	0.271	16	0	352	565	570
MZ2P6 (x1000)	2.127	25	1	39	628	564
MZ4P1 (x100)	1.859	3	1	52	112	17
MZ4P1 (x1000)	4.441	43	4	19	188	37
MZ4P4 (x100)	0.651	4	0	147	195	131
MZ4P4 (x1000)	2.412	40	0	43	787	432
LZ1P1 (x100)	0.567	35	0	60	114	52
LZ1P1 (x1000)	29.460	73	80	591	1638	848
LZ1P2 (x100)	1.119	890	198	1893	29551	36606
LZ1P2 (x1000)	0.895	35	0	1	60	89
LZ3P1 (x100)	1.345	4	0	257	431	199
LZ3P1 (x1000)	2.695	19	0	40	64	42
LZ3P2 (x100)	0.957	198	12	195	4041	4371
LZ3P2 (x1000)	3.150	302	22	293	5237	4962
LZ5P1 (x100)	0.766	113	3	101	4914	2270
LZ5P1 (x1000)	3.617	39	2	47	200	82
LZ5P2 (x100)	0.680	80	940	322	45629	40246
LZ5P2 (x1000)	4.940	1060	35	1	3373	2796
RP	0.004	42	27	1	51	46
RP (x10)	0.205	54	28	237	253	394
RP (x100)	1.288	147	33	80	260	384
RP (x1000)	0.927	74	26	1	89	164

Table IV. Tests with the maximum gamma adjustment rule

Problem	Initial Gap (%)	Fathomed Nodes	Reduced Nodes	*Best_Z* Node	Total Nodes	Time (CPUs)
MZ1P1 (x100)	0.663	1	42	1	358	56
MZ1P1 (x1000)	6.936	1	20	68	98	20
MZ1P4 (x100)	0.296	1	5	1	78	54
MZ1P4 (x1000)	2.150	1	12	65	122	88
MZ2P3 (x100)	1.085	0	50	181	707	285
MZ2P3 (x1000)	0.574	1	14	1	48	27
MZ2P6 (x100)	0.271	0	14	352	581	575
MZ2P6 (x1000)	1.262	1	44	25	388	331
MZ4P1 (x100)	1.834	1	11	46	114	18
MZ4P1 (x1000)	12.423	4	20	17	67	19
MZ4P4 (x100)	0.631	0	1	147	199	139
MZ4P4 (x1000)	2.296	2	7	125	401	258
LZ1P1 (x100)	0.567	1	32	60	148	61
LZ1P1 (x1000)	7.411	10	288	995	1387	872
LZ1P2 (x100)	0.982	0	736	2957	15297	18431
LZ1P2 (x1000)	0.895	1	32	1	94	101
LZ3P1 (x100)	1.337	1	28	221	338	156
LZ3P1 (x1000)	2.307	1	39	40	98	52
LZ3P2 (x100)	0.853	0	336	223	4135	4233
LZ3P2 (x1000)	3.187	11	330	456	3028	3015
LZ5P1 (x100)	0.685	1	284	225	5378	2451
LZ5P1 (x1000)	3.536	2	51	47	193	75
LZ5P2 (x100)	0.652	0	1717	456	29009	25306
LZ5P2 (x1000)	3.710	13	232	1	1662	1412
RP	0.004	1	35	1	92	109
RP (x10)	0.205	0	43	237	303	487
RP (x100)	0.845	1	75	80	234	311
RP (x1000)	0.927	1	71	1	162	195

- "*Best_Z* Node": the iteration at which the optimal solution was found;
- "Total Nodes": the total number of iterations required to explore the whole tree and prove optimality;
- "Time (CPUs)": the CPU time (in seconds) to compute bounds, generate nodes and perform the fathoming, feasibility and reduction tests.

The results shown in Tables III and IV illustrate the behavior described previously, and allow us to draw the following conclusions:

- When the minimum adjustment rule is used, the fathoming test F2 is very often active, while the number of reduced nodes is low, except for the RP problems. In these cases, this particular behavior may be explained by the relatively sparse nature of the RP network, which increases the chance of having inactive depots in a minimum cost flow solution. When the maximum adjustment rule is used, very few nodes are fathomed by criterion F2, while the number of reduced nodes is generally high.
- For some problems, the maximum rule may considerably reduce the size of the tree but, in the majority of the cases, the difference between the two rules does not appear to be significant.
- The CPU times are generally better or comparable when using the maximum rule. However, in the case of the RP problems, the minimum rule exhibits better CPU times for three of the four problems. This is due to the large number of reduced nodes generated in these cases.
- The optimal solution is generally found earlier by the minimum rule, but the difference is not significant. Indeed, the two rules identify the optimal solution

in less than 1000 iterations in all cases except one.

- As expected, the bounds computed by the maximum rule are generally tighter. However, the two rules are both fully satisfactory, since they produce initial gaps smaller than 1% in almost half of the cases and larger than 5% in only two or three cases.
- The algorithm is very sensible to a change in fixed costs. Indeed, when experimenting with low or high fixed costs, one of the components of the problem, minimum cost network flow or location, dominates the other and the solution is generally easily found. However, when the fixed costs take "intermediate values", the enumeration trees grow considerably, especially for large-sized problems. In these cases, the maximum adjustment rule is clearly preferable.

As a general conclusion, we recommend the use of the maximum adjustment rule, since it is competitive in all cases and it outperforms the minimum rule when used to solve "hard" problems.

We finally compare the results of our algorithm with those obtained by other methods, in order to assess its competitiveness. Table V shows the value of the best solution ("$Best_Z$"), rounded in thousands of units, and the total CPU time ("Time (CPUs)"), on a SUN Sparc2 workstation, for the following methods:

- "DUAL": the dual-ascent heuristic of Crainic and Delorme (1992), which corresponds to solve only the root node of the tree;
- "TABU": the Tabu search heuristic of Crainic, Gendreau, Soriano and Toulouse (1992);

Table V. Comparison with other methods

Problem	TABU		BBWR		DUAL		BBFF	
	Best_Z	Time (CPUs)	*Best_Z*	Time (CPUs)	*Best_Z*	Time (CPUs)	*Best_Z*	Time (CPUs)
MZ1P1 (x100)	42499*	194	42593	137	42499*	1	42499*	56
MZ1P1 (x1000)	61565*	176	61565*	126	65320	5	61565*	20
MZ1P4 (x100)	109177	889	109285	545	108914*	3	108914*	54
MZ1P4 (x1000)	142063*	718	142063*	511	143036	17	142063*	88
MZ2P3 (x100)	69144*	609	69430	330	69487	6	69144*	285
MZ2P3 (x1000)	88518*	498	88518*	336	88518*	16	88518*	27
MZ2P6 (x100)	248308	1333	248703	800	248329	3	248178*	575
MZ2P6 (x1000)	291199	1050	290831	701	292080	45	290271*	331
MZ4P1 (x100)	21134*	195	21134*	138	21406	1	21134*	18
MZ4P1 (x1000)	46014*	183	46014*	129	50011	6	46014*	19
MZ4P4 (x100)	93874*	802	93874*	552	94216	9	93874*	139
MZ4P4 (x1000)	136642*	727	136642*	454	137668	22	136642*	258
LZ1P1 (x100)	23286*	340	23375	349	23349	5	23286*	61
LZ1P1 (x1000)	42130	330	45013	274	43909	52	42062*	665
LZ1P2 (x100)	65495	1103	66617	942	65308	14	65308	1136
LZ1P2 (x1000)	89827	734	89068*	855	89068*	25	89068*	101
LZ3P1 (x100)	33271	396	33400	426	33601	7	33267*	156
LZ3P1 (x1000)	56832*	366	56832*	375	57553	26	56832*	52
LZ3P2 (x100)	61174	922	62445	794	61052	20	60959*	1021
LZ3P2 (x1000)	101278	678	102828	658	101420	59	100851*	915
LZ5P1 (x100)	—	—	49352	335	48917	7	48860*	441
LZ5P1 (x1000)	—	—	78480	330	77589	12	75903*	75
LZ5P2 (x100)	—	—	96666	677	96312	16	96114*	823
LZ5P2 (x1000)	—	—	133605	599	133454*	18	133454*	791
RP	—	—	53229	1584	53176*	5	53176*	109
RP (x10)	—	—	56829	1398	56526	9	56515*	487
RP (x100)	—	—	72460	1287	72385	41	72339*	311
RP (x1000)	—	—	166506*	1194	166506*	32	166506*	195

- "BBWR": a branch-and-bound approach based on a *weak relaxation* of the problem, as in Crainic, Delorme and Dejax (1992). Note that we considerably enhanced their basic algorithm by adapting the techniques introduced in Section II of the present paper (see the Appendix for a description of these enhancements);
- "BBFF": the present algorithm.

The two branch-and-bound algorithms were executed up to a maximum of 1000 iterations, while the TABU Search procedure was allowed to run for 300 iterations. Note that the BBWR algorithm could not complete the enumeration for any test problem.

The results show that the branch-and-bound algorithm completely outperforms the other methods, both in solution quality (an asterisk indicates an optimal solution) and computing times. Indeed, for some cases, it may run ten times faster than the BBWR algorithm and still find a better solution. Moreover, when it takes more time, it always identifies a strictly better solution.

Another important question is whether the extra effort required by the algorithm, compared to the dual-ascent heuristic, is rewarded by a significant improvement in the quality of the solutions. This is the case, indeed. On the one hand, only some 25% of the problems are solved to optimality by the heuristic. On the other hand, improvements greater than 1% (sometimes as much as 8%) are obtained for some other 25% of the problems.

CONCLUSION

We have presented a branch-and-bound algorithm for solving the multicommodity location problem with balancing requirements. The bounds are computed by a dual-ascent procedure derived from Lagrangean relaxations of the problem. We particularly emphasized the development of techniques to reduce the size of the tree and to save on computing time. These include strong fathoming criteria, and branching rules which use the dual information to decide which subproblem should be examined next.

Experimental results have shown that the choice of a branching rule is critical. Indeed, we demonstrated that one of the dual rules is clearly superior to the others and significantly reduces the size of the tree. We have also shown that the algorithm performs well on a wide variety of problems, including an actual large-scale application. Furthermore, on these problems, the algorithm completely outperforms other methods proposed in the literature.

REFERENCES

Aikens, C.H. (1985), *Facility Location Models for Distribution Planning*, **European Journal of Operational Research** 22, 263–279.

Balakrishnan, A., Magnanti, T.L. and Wong, R.T. (1989), *A Dual-Ascent Procedure for Large-Scale Uncapacitated Network Design*, **Operations Research** 37(5), 716–740.

Cornuejols, G., Nemhauser, G.L. and Wolsey, L.A. (1990), *The Uncapacitated Facility Location Problem*, in **Discrete Location Theory**, R.L. Francis and P.B. Mirchandani (editors), Wiley-Interscience, 119–168.

Crainic, T.G., Dejax, P. and Delorme, L. (1989), *Models for Multimode Multicommodity Location Problems with Interdepot Balancing Requirements*, **Annals of Operations Research** 18, 279–302.

Crainic, T.G. and Delorme, L. (1992), *Dual-Ascent Procedures for Multicommodity Location Allocation Problems with Balancing Requirements*, forthcoming, Transportation Science.

Crainic, T.G., Delorme, L. and Dejax, P. (1992), *A Branch-and-Bound Method for Multicommodity Location with Balancing Requirements*, forthcoming, European Journal of Operational Research.

Crainic, T.G., Gendreau, M., Soriano, P. and Toulouse, M. (1992), *A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements*, forthcoming, Annals of Operations Research

Erlenkotter, D. (1978), *A Dual-Based Procedure for Uncapacitated Facility Location*, **Operations Research** 26(6), 992–1009.

Geoffrion, A.M. (1974), *Lagrangean Relaxation for Integer Programming*, **Mathematical Programming Study** 2, 82–114.

Grigoriadis, M.D. and Hsu, T. (1979), *RNET – The Rutgers Minimum Cost Network Flow Subroutines*, Rutgers University, New Brunswick, New Jersey.

Ibaraki, T. (1987), *Enumerative Approaches to Combinatorial Optimization*, **Annals of Operations Research** 10–11.

Krarup, J. and Pruzan, P.M. (1983), *The Simple Plant Location Problem: Survey and Synthesis*, **European Journal of Operational Research** 12, 36–81.

Nemhauser, G.L. and Wolsey, L.A. (1988), **Integer and Combinatorial Optimization**, Wiley-Interscience.

Van Roy, T.J. and Erlenkotter, D. (1982), *A Dual-Based Procedure for Dynamic Facility Location*, **Management Science** 28(10), 1091–1105.

APPENDIX – WEAK RELAXATION APPROACH

In this Appendix, we present an improved version of the branch-and-bound approach proposed by Crainic, Delorme and Dejax (1992). The improvements are mainly inspired by the ideas developed in Section II of the present paper.

The bounding procedure is based on a *weak formulation* of the problem, in which constraints (4a) and (4b) of problem P1 are aggregated:

$$\sum_{p \in P} \left(\sum_{i \in C_1^j(p)} x_{ijp} + \sum_{i \in C_2^j(p)} x_{jip} \right) \leq \Delta_j y_j \qquad \forall j \in D \qquad \text{(A1)}$$

The *weak relaxation* is then obtained by replacing the binary decision variables with positive continuous ones. Then, constraints (A1) must hold with equality in any optimal solution of the relaxation. Thus, we can drop the continuous decision variables and obtain the following problem:

Problem PA1

Minimize

$$\sum_{p \in P} \left\{ \sum_{(i,j) \in E_1(p)} (c_{ijp} + f_j/\Delta_j) x_{ijp} + \sum_{(j,i) \in E_2(p)} (c_{jip} + f_j/\Delta_j) x_{jip} + \sum_{(j,k) \in E_3(p)} s_{jkp} w_{jkp} \right\} \qquad \text{(A2)}$$

Subject to (2a), (2b), (3), (5a), (5b), (6).

This is a multicommodity uncapacitated minimum cost flow problem, which decomposes into $|P|$ single-commodity minimum cost flow problems. In fact, it is a particular case of problem P4, in which the γ multipliers associated to a depot

j take all the same value and no slack is allowed in the dual constraints (11). An upper bound is also computed in the same way as for problem P4.

Note that this lower bound is tighter than the one proposed by Crainic, Delorme and Dejax (1992), since they use Δ, the sum of *all* supplies and demands, instead of Δ_j.

The techniques presented in Section II may be applied, since we have the analogous of Proposition 2:

Proposition A1.

Let S be a subproblem selected at step 2 of procedure BB and Z^L its lower bound obtained by solving the weak relaxation. Let S_1 be the 1-son node of S and $Z(S_1)$ the optimal value of S_1. We then have

$$Z^L + f_{j^*}(1 - y_{j^*}) \leq Z(S_1) \qquad \text{(A3)}$$

Thus, the fathoming criteria, the feasibility test and the reduction test may be easily adapted in this context.

We also tested two sets of branching rules, already proposed by Crainic, Delorme and Dejax (1992). One set of rules is based on a *difference* measure:

$$D_j = f_j(1 - y_j) \qquad \forall j \in D_{01}(S) \qquad \text{(A4)}$$

The other set uses a *quotient* measure:

$$Q_j = y_j / f_j \qquad \forall j \in D_{01}(S) \qquad \text{(A5)}$$

We concluded that the quotient rule which selects first a 0-son node is clearly superior to the others. Consequently, this rule was used in the tests reported in Section III.